

PALLAS

Z 80 Fejlesztőrendszer
(Felhasználói kézikönyv)

1. Bevezetés	3
2. Képernyős szövegszerkesztő	4
2.1. A szerkesztő állapotai	4
2.2. Pozicionálás a szövegben	4
2.2.1. Az összes állapotban érvényes parancsok	4
2.2.2. Csak "Command" állapotban érvényes parancsok	5
2.2.3. A parancsok automatikus ismétlése	6
2.3. Szövegjavító parancsok	6
2.3.1. A szövegben beírható karakterek	6
2.3.2. Minden állapot érvényes javító parancsok	6
2.3.3. Csak "Command" állapotban érvényes parancsok	7
2.4. Egyéb parancsok	8
2.5. Tartalom szerinti keresés	9
2.5.1. Parancsok	9
2.5.2. A mintaillesztés menete	9
2.6. Bonyolultabb parancsok megtanítása	10
3. A Z 80 mikroprocesszor általános leírása	11
3.1. Belső felépítés	11
3.2. Regiszterkészlet	11
3.3. Utasításkészlet	13
3.3.1. Címzés módok	13
3.3.2. Adatmozgató utasítások	15
3.3.3. Aritmetikai és logikai utasítások	22
3.3.4. CPU vezérlő utasítások	28
3.3.5. Forgató és eltoló utasítások	29
3.3.6. Bitműveletek	31
3.3.7. Ugróutasítások	32
3.3.8. Szubrutinhívó és visszatérő utasítások	33
3.3.9. Be- és kiviteli utasítások	34
3.3.10. A jelzők beállítása	36
4. A Z 80 mikroprocesszor programozása ASSEMBLY nyelven	39
4.1. Az ASSEMBLY nyelv felépítése	39
4.1.1. Direktívák	39
4.1.2. Utasítások	41
4.1.3. Aritmetikai kifejezések	41
4.1.4. Fordítást vezérlő speciális megjegyzések	41
4.2. Az ASSEMBLER használata	42
4.2.1. Az ASSEMBLER elindítása	42
4.2.2. Hibajelzések	43

5. Futtató-belövő: MONITOR	44
5.1. Regiszterkezelő és futtató	44
5.2. Visszafejtő program	46
5.3. Memoria megjelenítő	47
5.4. Szimbolikus belövési lehetőségek	48

1. Bevezetés

A PALLAS egy program fejlesztőrendszer a PRIMO számítógépre. Segítségével a Z 80 mikroprocesszor gépi nyelvén írhatunk programokat, ami lehetővé teszi a PRIMO adottságainak maximális kihasználását. Ugyanakkor a BASIC számára megmaradó területen a program intelligensebb részeit BASIC nyelven is megírhatjuk, ha ezen programrészek lassúsága nem okoz problémát. A PALLAS a képernyős szövegszerkesztőből, az ASSEMBLER-ből és a MONITOR-ból áll. A képernyős szövegszerkesztő segítségével lehetőség nyílik a gépi kódú programok forrásainak kényelmes begépelésére, javítására. Az így kapott, szimbólikus formában megadott utasításokat az ASSEMBLER fordítja le a PRIMO számára is érthető kódokká, s ekközben a formailag hibás utasításokról hibajelzések révén kapunk információkat. A formailag hibátlan programokat a MONITOR segítségével tudjuk tesztelni, a logikai hibáktól megtisztítani. Ehhez a munkához a MONITOR olyan funkciói nyújtanak hatékony segítséget mint például a lépésenkénti futtatás lehetősége, vagy a memória tartalmának visszfejtése gépi kódú utasításokká. A fentieken kívül a kézikönyv külön fejezete foglalkozik a Z 80 mikroprocesszor utasításainak leírásával, amely fejezetet a Z 80 gépi kódját már ismerők kihagyhatják.

2. Képernyős szövegszerkesztő

2.1. A szerkesztő állapotai

A program elindítása után azonnal a képernyős szövegszerkesztő jelentkezik be. Ilyenkor a legfelső sorban egy > jel, a következő sorban pedig egy <EOF> felirat található. A későbbiekben (ha már gépeltünk vagy betöltöttünk valamit) a sorok elején egy > jel található azért, hogy meg tudjuk különböztetni a sorok elejét a 40 karakternél hosszabb sorok folytatásától, mivel a folytatólagos részek előtt nincs > jel. A szöveg végén a > jel helyett <EOF> felirat áll.

A szövegbe bevihetünk betűket, számokat, írásjeleket, betűköz karaktereket. Ezek egy pozícionyi helyet foglalnak el. A sorhatároló karakter a RETURN, melyet a szövegszerkesztő outputjában a 0AH hexadecimális kód képvisel. Ez az ASCII karakterkészletben az LF (line feed) karakter. A mezőhatároló karakter a SHIFT+jobbra nyíl, melyet a szövegszerkesztő outputjában a 09H hexadecimális kód ábrázol. Ez az ASCII karakterkészlet TAB karaktere. A sorhatároló karakter után a sor véget ér, a következő sor előtt egy > jel jelenik meg. A mezőhatároló karakter hatására a következő (a sorban utána álló) karakter a következő nyolccal osztható pozícióban jelenik meg, a hézagot betűközök töltik ki. Ezzel elérhetjük, hogy az azonos programmezők egymás alatt helyezkedjenek el anélkül, hogy számolnunk kellene, hogy hány betűközöt üssünk le az előző mező után. A mezőhatároló karakter tehát legalább 1, de legfeljebb 8 pozíciót foglal el.

A szövegszerkesztőnek több állapota van, s a klaviatura vezérlő billentyűinek korlátozott száma miatt ugyanannak a vezérlő billentyűnek más-más szerep jut az egyes állapotban. A képernyő utolsó sorában mindig kiíródik az aktuális állapot neve is. Ezek a következők lehetnek:

Command

A látható karaktereket (betűket, számokat, írásjeleket) megpróbálja parancsként értelmezni. Az "Insert" állapotba "I", az "Over" állapotba "O" betű leütésével térhetünk át.

Insert

A leütött betű, szám vagy írásjel a kurzor pozíciója elé beíródik, a kurzor pedig eggyel tovább lép, végeredményben tehát ugyanarra a karakterre fog mutatni, mint az előbb, de előtte már ott látható az általunk leütött karakter. A "Command" állapotba BRK leütésével mehetünk. Az "Over" állapotba csak a "Command" állapoton keresztül juthatunk.

Over

A leütött karakter átírja az ott talált karaktert, kivétel a mezőhatároló és a sorhatároló karakter. Ezek mindig az adott karakter elé beíródnak, ugyanúgy, mintha "Insert" állapotban dolgoznánk. Ha a gépelés során mezőhatároló vagy sorhatároló karaktert akarunk felülírni, akkor is ugyanez a helyzet, tehát nem fordulhat elő, hogy tévedésből sorokat összemásoljunk a sorhatároló karakter átírásával. A "Command" állapotba BRK leütésével mehetünk. Az "Insert" állapotba "Command" állapoton keresztül juthatunk.

2.2. Pozicionálás a szövegben

2.2.1. Az összes állapotban érvényes parancsok

A szöveg különböző részeit sokféleképpen elérhetjük, először a minden állapotban működő lehetőségeket soroljuk fel:

Balra nyíl

Egy karaktert visszalép. Ha mezőhatároló karakter jön, akkor azt teljesen átlépi. Hasonlóan a sorhatároló karakter után is az előző sor végére áll. Ha a szöveg legelején próbáljuk meg, akkor "Begin of file" hibajelzést kapunk.

Jobbra nyíl

Egy karaktert előre lép. Ha mezőhatároló karakter jön, akkor azt teljesen átlépi. Hasonlóan a sorhatároló karakter után is a következő sor elejére áll. Ha a szöveg legvégén próbáljuk meg, akkor "End of file" hibajelzést kapunk. A hibajelzések csak tájékoztató jellegűek, a következő karakter leütése után eltűnnek.

Lefelé nyíl

Egy sort lefelé lép. Megpróbál a következő sor ugyanolyan pozíciójába lépni. Ha ez nem sikerül, mert mezőhatároló bájt vagy sor vége után kellett volna állni, akkor a mezőhatároló elé vagy a sor végére áll. A parancs az eredetileg kijelölt pozíciót veszi célba akár több soron keresztül is addig, amíg csak lefelé léphetünk. Így pl. ha egy hosszú sor végéről egy másik hosszú sor végére csak ilyen parancsok ismétlésével lépünk át, akkor is odatalál, ha közben voltak rövidebb sorok. Ha az utolsó sor utánra próbálunk meg lépni ezzel a paranccsal, akkor "End of file" hibajelzést kapunk.

Felfelé nyíl

Egy sort felfelé lép. Megpróbál az előző sor ugyanolyan pozíciójába lépni. Ha ez nem sikerül, mert mezőhatároló bájt vagy a sor végére áll. Ez a parancs szintén az eredetileg kijelölt pozíciót igyekszik elérni, akár több soron keresztül is addig, amíg csak felfelé léphetünk. Amennyiben a legelső sor elé próbálunk meg lépni ezzel a paranccsal, úgy "Begin of file" hibajelzést kapunk.

SHIFT+lefelé nyíl (Billentyűs gépen SHIFT + i)

A sor végére áll. Ha a sor hosszabb mint 255 karakter, akkor nem találja meg a végét, ezért lehetőleg kerüljük az ilyen hosszú sorok használatát.

CTR+'S'

A sor elejére áll.

CTR+'G'

A következő nyolccal osztható pozícióra ugrik. Ha hosszabb a sor mint 255 karakter, a 256. helyett a legelsőbe fog állni. Ha a sor végére ér, nem lép tovább.

CTR+'N'

A következő sor elejére áll.

2.2.2. Csak "Command" állapotban érvényes parancsok

A következő parancsok csak a "Command" állapotban érvényesek, a többiben a szövegbe kerülnek az itt leírt karakterek.

'<'

A szöveg legelejére áll. Ha már eddig is ott volt, akkor "Begin of file" hibajelzést kapunk.

'>'

A szöveg legvégére áll. Ha már eddig is ott volt, akkor "End of file" hibajelzést kapunk.

'-'

Felfelé lép 10 sort. Kaphatunk "Begin of file" hibajelzést.

'+'

Lefelé lép 10 sort. Kaphatunk "End of file" hibajelzést.

'*'

Egy könyvjelzőt helyez el. Az így megjelölt pozícióra a ':' segítségével ugorhatunk. A könyvjelzőt bizonyos parancsok is áthelyezik, ezekre felhívjuk a figyelmet.

','

Számjelzésű könyvjelzőt definiál. A legalsó sorban egy "Set marker:"

felíratot kapunk, erre egy {0..9} tartományba eső számmal kell válaszolnunk. A későbbiekben erre a pontra a ',' paranccsal ugorhatunk. Néhány parancs automatikusan áthelyezi némelyik könyvjelzőt, ezekre külön felhívjuk a figyelmet.

','

Ezzel a paranccsal ugrunk a számjelzésű könyvjelzőre. A legalsó sorban "Goto marker:"

felíratot kapunk, erre egy {0..9} tartományba eső számmal kell válaszolnunk. Ekkor a megjelölt helyre ugrik a kurzor. A könyvjelzők a szerkesztés elején a szöveg elejére mutatnak.

2.2.3. A parancsok automatikus ismétlése

A Command állapotban leütött számjegy hatására a legalsó sorban egy Repeat:

felírat jelenik meg, ami után a leütött számot írja ki a szerkesztő, majd egy egyéb parancs bebillentyűzésével befejezzük. Az ekkor megadott parancsot annyszor fogja végrehajtani egymás után, amekkora számot begépelünk. Ezzel könnyen megadhatunk pl. olyan utasításokat, hogy keresse meg a szöveg 200.sorát. ('<','1','9','9', Lefelé nyíl)

Az "Insert" vagy "Over" állapotban egy CTR+'R' hatására jelenik meg a Repeat:

felírat, ami után a számot és a parancsot gépelhetjük. Ezzel pl. beírhatunk egyszerűen 30 db. '*' karaktert a programrészek elválasztására. lasztására. (CTR+'R', '3', '0', '*')

2.3. Szövegjavító parancsok

2.3.1. A szövegbe beírható karakterek

A szövegbe csak "Over" vagy "Insert" állapotban vihetünk be karaktereket, melyek a mezőhatároló és sorhatároló karakteren kívül csak betűk, számok vagy írásjelek lehetnek. Az eltévesztett betűket SHIFT + balra nyíl segítségével törölhetjük.

2.3.2. Minden állapotban érvényes javító parancsok

A következő egyszerű javító parancsok minden állapotban működnek:

SHIFT+Balra nyíl

A kurzor előtt álló karakter törlése. Ha a szöveg elején állunk, "Begin of file" hibajelzést kapunk.

CTR+'C'

Az utoljára SHIFT+Balra-val törölt karaktert visszaállítja. Ezt többször is megtehetjük. A javítás elején (a legelső törlésig) egy betűköz karaktert ír vissza.

CTR+'O'

Az adott sor elé beszúr egy üres sort és az elejére áll.

CTR+'P'

Az adott sort teljesen törli.

2.3.3. Csak "Command" állapotban érvényes parancsok.

A következő javító parancsok csak "Command" állapotban működnek.

'/'

Blokk elejének kijelölése. A szerkesztő képes nagyobb szövegblokkok kezelésére is. A blokk elejét ezért meg kell jelölni. Erre a pontra azután a ';' leütésével közvetlenül is ugorhatunk.

'C'

A '/' megjelölt pozíciótól a kurzorig terjedő blokk kivágása a szövegből (a kurzor által mutatott karakter már nem számít bele). Az így kivágott blokk nem vész el, mert egy átmeneti tárolóban megőrzi. Ugyanide kerül a CTR+'P' paranccsal kivágott sor is. Ennek az átmeneti tárolónak a tartalmát tetszőleges helyre beszúrhatjuk az 'R' paranccsal. Ezzel egyszerűen megoldható szövegrészek másolása, duplázása. Ha nem jelöltük ki a blokk elejét, vagy az elejét jelző mutató hátrább van, mint a kurzor, akkor "No begin" hibajelzést kapunk.

'R'

Szöveg beszúrása az átmeneti tárolóból a kurzor által mutatott pontra. Ha éppen nincs semmi az átmeneti tárolóban, akkor "No text" hibajelzést kapunk. A beszúrást tetszőlegesen sokszor megismételhetjük.

'X'

A következő szóban a nagybetűket kisbetűre váltja.

'Y'

A következő szóban a kisbetűket nagybetűkre váltja.

'A'

A memóriában található szöveg legvégére beszúr egy magnószalagon található szöveget, amit előzőleg már 'W' paranccsal kivittünk. A szöveg nevét az első sorban megjelenő

Name: (*)

felirat után kell gépelni, majd a végén egy RETURN leütésére kezdi meg a kért szöveg keresését a magnón. Ha a szöveg megvan, a PRIMO LOAD parancsához hasonlóan kiírja a blokkok és a hibás blokkok számát. Ha nem fér be a memóriába a szöveg, akkor "Not enough memory" hibajelzést kapunk. A keresést RESET megnyomására szakíthatjuk félbe, amire az előző szöveg jelenik meg a képernyőn. Ha a régi szöveget nem akarjuk, ki kell törölni a betöltés előtt. ('<', '/', '>', 'C', 'C'.)

'W'

A teljes szöveget kiviszi magnetofonra. A kiírt szöveget, a 'V' paranccsal ellenőrizhetjük le.

{Name: (*) }

'V'

A 'W' paranccsal kiírt file ellenőrzése. A hibák számára a legalsó sorba írt hibás rekordok számából következtethetünk.

{Name: (*) }

'T'

A teljes szöveget kiírja az ASSEMBLER számára érhető formában. Ilyenkor minden sor külön blokkba kerül és a szöveg kétszer megy ki, hogy az ASSEMBLER közvetlenül magnóról le tudja fordítani. Ezt nagyon hosszú programoknál használhatjuk, ha a szöveg és a belőle készült program már nem fér el a rendelkezésre álló memóriában. A használatáról bővebbet az ASSEMBLER leírásánál találhatunk.

{Name: (*) }

(*) Csak az első 10 karaktert veszi figyelembe!

Ha magnó helyett floppy-t használunk, először be kell tölteni a C DOS-t és utána a PALLAS-t. A fent felsorolt parancsok értelemszerűen érvényesek. Az alsó sorban megjelenő

Name:

után a nevet :-tal kezdjük. (Ez utal a floppy-ra)

2.4. Egyéb parancsok.

A most következő parancsok nem szigorúan a javításhoz tartoznak, de segítik a programozó munkáját. Csak "Command" állapotban működnek.

'Z'

Az ASSEMBLER elindítása. Az alsó sorban megjelenik egy

Options:

felirat, a bővebb ismertetést majd az ASSEMBLER leírásánál adjuk meg.

'Q'

Kilépés a BASIC-ba. Az alsó sorban egy

Quit (Y/N):

felirat jelenik meg, amire ha Y-nal válaszolunk, akkor a BASIC jön be, ha bármi mással, akkor a szerkesztőben marad. A BASIC-ből RESET megnyomásával térhetünk vissza a szerkesztőbe.

'K'

A szöveg törlése és új maximális méret megadása. Az alsó sorban egy

Ramtop:

felirat jelenik meg, amire egy hexadecimális számmal, vagy RETURN-nel kell válaszolnunk. A megadott számmal lesz a határ a lefordított program és a szöveg rendelkezésére álló hely között. (csak RETURN esetén nem változik) A rendelkezésre álló memóriaterület a PALLAS program fölött helyezkedik el, és a képernyő területéig tart. Alap értelmezésben ez 1:3 arányban oszlik el az ASSEMBLER által előállított gépi kód és a forrásszöveg között. A MONITOR-ban belépve az SP mutat a gépi kódú program rendelkezésre álló terület végére, a PC az elejére. Használatát csak gyakorlottabb programozóknak, nagyobb programok írására ajánljuk.

'M'

Belép a MONITOR-ba. Visszalépni 'Q' megnyomásával lehet. A MONITOR kezelését a későbbiekben részletezzük.

'B'

A rendelkezésre álló szabad terület kiírása. Az alsó sorban a következőbillentyű megnyomásáig az

xxxxx bytes free

felirat jelenik meg, ahol xxxxx a rendelkezésre álló hely. Ennyi karaktert vihetünk még be a szövegbe, de vigyázzunk, mert az ASSEMBLER-nek is kell még terület a fordításhoz, és ha teljesen kitöltjük a memóriát, akkor biztosan nem tudjuk lefordítani a programunkat.

2.5. Tartalom szerinti keresés

2.5.1. Parancsok

A szerkesztő segítségével nemcsak egyenesen az ismert pozíciókra állhatunk rá, hanem a szövegben tartalom szerinti keresésre is mód nyílik. A "Command" állapotban leütött 'P' hatására az alsó sorban megjelenik vagy az "Insert" vagy az "Over" állapot jelzője, valamint azután egy "Modell" felirat, a képernyő pedig üres lesz, vagy az előző minta jelenik meg rajta. A mintát ugyanúgy szerkeszthetjük meg, mint a normál szöveget, majd "Command" állapotban leütött 'D' vagy 'U' hatására a program visszatér a normál szöveghez, és elkezd keresni a mintára illeszkedő karakter sorozatot. A 'D' hatására kurzortól indulva a szöveg vége felé, az 'U' hatására a kurzortól indulva a szöveg eleje felé. Ha megtalálta akkor a kurzor a mintával azonos szövegrész elejére áll, ha nem, akkor "Not found" hibajelzést kapunk. A keresési folyamat bármely billentyű megnyomására abbamarad. Az alsó sorban "Break" hibajelzés kerül. A keresés előtt az 1. könyvjelző a kurzor pozíciójára áll be, így oda könnyen visszatalálhatunk. Ha a keresés sikeres volt, akkor az illeszkedő rész elejére állítja a 2. könyvjelzőt. (csak egy minta létezik, lehet több sor is!)

A 'D' vagy 'U' parancsokon kívül még van egy 'F' parancs is, ami a legutoljára kiadott 'D' vagy 'U' parancs által meghatározott irányban fog keresni. A 'D', 'U' és 'F' parancsokat kiadhatjuk "Command" állapotban anélkül, hogy a 'P' paranccsal a mintát megváltoztatnánk. A keresés mindig a kurzortól indul. Így találhatjuk meg pl. az 5. mintára-illeszkedő sort ('5', 'F'). "Insert" vagy "Over" állapotban az 'F' parancs megfelelője a CTR+'F'.

A 'P', 'D', 'U' parancsok csak "Command" állapotban működnek.

A szisztematikus javításokat könnyíti meg az 'S' parancs, ami megkeresi a mintára illeszkedő karakter sorozatot, kivágja, majd helyére illeszti az átmeneti tárolóban lévő ('C' vagy CTR+'P' parancs hatására odakerült) szöveget. Ezt szintén kiadhatjuk a minta megváltoztatása után vagy anélkül.

2.5.2. A mintaillesztés menete.

A szerkesztő a mintát a kurzortól kezdi illeszteni, karaktert karakterre, amíg eltérést vagy minta végét nem találja. Ha a minta végéhez ért, akkor a keresés sikeres volt. Ha eltérést talál, akkor egyet lép a kurzor és a fent leírt folyamat előről kezdődik.

Ha eléri a szöveg elejét vagy végét, akkor a keresés sikertelen, ekkor kapjuk a "Not found" hiba jelzést. A mintában nem csak normál karakterek lehetnek, hanem egyéb Joker karakterek, illetve szövegrészek is. A következő jelölésekkel érhetjük ezt el

%! Joker karakter

%? Joker szövegrész.

A Joker karakter minden karakterre illeszkedik, a sor végét és a szöveg végét kivéve. A Joker szövegrész tetszőleges hosszú szövegre illeszkedik (lehet akár 0 is), de nem foglalhat magába sorhatároló karaktert. Ezzel adhatjuk meg pl. hogy keressük az A regiszterbe tetszőleges címről

betöltő utasítást. A minta ehhez a következő módon néz ki.

LD A, (%)

A mintában egymás mellett álló betűkötők azt a minimális számot határozzák meg, amennyinek ott egymás mellett állnia kell. Ha ennél több van ott, az is illeszkedésnek számít. Azaz, ha a szerkesztő az illesztéskor betűkötőket talál a mintában, miután mindnek talált párt a szövegben és a következő mintakarakter már nem betűkötő, akkor a szövegben is átugorja őket a következő egyéb karakterig.

A minta állhat több sorból is, így kereshetünk bonyolultabb program részeket is.

2.6. Bonyolultabb parancsok megtanítása

A szerkesztő képes megtanulni bármilyen parancssorozatot, és azt tetszés szerint megismételni. Ennek segítségével a program hasonló részeinek gépelését egyszerűbben oldhatjuk meg. A

CTR+'B'

Parancs hatására a legalsó sor jobb oldalán megjelenik egy

Learn:

felirat, ami jelzi, hogy az ezután következő karakterek tárolódnak. A CTR+'B' újbóli megnyomása visszaállítja a normál állapotot. Az újra lenyomott CTR+'B' nem törli a tárolt parancsokat, hanem a következőt írja. A végrehajtáskor emlékszik arra, hogy hol függesztettük fel a tanulást. A "Learn" üzemmódban ültött

CTR+'E'

azonban minden megjegyzett karaktert töröl, így módon újra kezdetjük a parancsok tanítását. A normál üzemmódba visszalépve a CTR+'D' parancs segítségével indíthatjuk el a végrehajtást. Ekkor az alsó sorban "Run" felirat kerül oda, ahol a tanuláskor a "Learn"-t láttuk. A CTR+'D' parancsot is kiadhatjuk ismétléssel, ez nem zavarja a megtanult részekben belül ismétlések pontos elvégzését.

3. Z 80 mikroprocesszor általános leírása

3.1. Belső felépítés

a Z 80 mikroprocesszor a programozó számára legfontosabb részei az aritmetikai és logikai egység (ALU) valamint a regiszterkészlet. Az aritmetikai és logikai egység végzi a műveleteket, az eredményeket a regiszterekben és a memóriában tárolhatjuk. A regiszterek is tulajdonképpen memóriák, de a mikroprocesszoron belül foglalnak helyet, és ezért az elérésük sokkal egyszerűbb és gyorsabb. A regiszterek lehetnek egy vagy két bájt, azaz 8 vagy 16 bit nagyságúak. Bizonyos műveletekben két 8 bites regiszter együtt alkot egy 16 bites regisztert, amit ezért regiszterpárnak nevezünk.

3.2. Regiszter készlet

A Z 80 CPU (központi feldolgozó egység) 26 bájt írható-olvasható memóriát tartalmaz, melynek 8 vagy 16 bites csoportjai a regiszterek.

PC

A programszámláló regiszter, a soron következő utasításra mutat. A CPU ebből tudja meg, mi lesz a soron következő utasítás. A 16 bites mérete miatt a maximális memóriakiépítés 64 KB (1 KB = 1024 B, bájt)

SP

A veremmemória mutatója. A veremmemória jellegzetessége, hogy az utoljára beírt adatot olvashatjuk ki először. Ezt a szervezést a SP segítségével valósítja meg a mikroprocesszor. A verembe beíró utasítás először csökkenti kettővel az SP tartalmát, majd az általa mutatott címre beírja az elmenteni kívánt két bájt. Kiolvasáskor először kiolvasuk az SP által mutatott címről két bájt, majd megnöveljük kettővel az SP tartalmát. Ezzel elérjük, hogy mindig az utoljára beírt adat kerül elő legelőször. A verem segítségével nagyon hatékonyan megoldhatjuk a regiszterek értékének átmeneti tárolását, és a szubrutinhívásokat.

IX

Index regiszter, egy 16 bites un. báziscímet tárol. Az indexelt címezést használó utasításban még található egy 8 bites előjeles szám (a { - 128 .. 127 } tartományban), a kettő összege mutat az utasítás operandusára. Nagyban megkönnyíti sok táblázattal dolgozó programok írását.

IV

Ugyanolyan index regiszter, mint az IX.

I

Megszakítási regiszter. A Z 80 mikroprocesszor képes a megszakítások kezelésére is. A megszakítás azt jelenti, hogy a CPU egy külső esemény hatására az éppen futó programot abbahagyja, és rátér a külső eseményhez kapcsolódó (sürgősebb) feladat megoldására. Ekkor egy szubrutinhívást végez a memória egy bizonyos helyéről, és az I regiszter ennek a címnek kiszámításához szükséges.

R

Memóriafrissítési regiszter, a dinamikus memóriák használatához kell. A tartalma minden utasítás végrehajtás után eggyel nő, és az utasítás beolvasásakor a CPU elküldi a tartalmát a dinamikus memóriákhoz. Ezzel el lehet érni, hogy a memóriák ne felejtsek el a tartalmukat. A programozónak nincs rá nagy szüksége, esetleg véletlen számként használhatjuk fel a tartalmát.

A

Akkumulátor, az aritmetikai és logikai műveletek nagyrésze e regiszter és egy másik operandus között játszódik le, az eredmény is ide kerül.

F

Az aritmetikai műveletekről tartalmaz információkat, a legfontosabb bitjei a Z (zéró), C (átvitel), S (előjel) és a P/V (párosság vagy túlcsoordulás) jelzők, amelyek feltételes utasításokkal vizsgálhatók. Az utasítások különbözőképpen állítják be ezeket a jelzőket, erre később részletesen kitérünk.

B általános célú regiszter. 8 bit tárolására képes.

C általános célú regiszter.

D általános célú regiszter.

E általános célú regiszter.

H általános célú regiszter.

L általános célú regiszter.

Az általános célú regiszterek kettosével regiszterpárokat képeznek, melyeket BC, DE, HL regiszterpároknak nevezzük. Bizonyos utasítások ezeket 16 bites adatként kezelik. Regiszterpárt képez még az AF is.

Az általános célú regiszterekből, az akkumulátorból és a jelzőből két készlet van; egy fő és egy második készlet, melyeket megfelelő utasításokkal cserélhetünk egymás között.

A regiszterkészlet vázlatos felépítését a következő ábrán láthatjuk:

Fő regiszterkészlet

Második regiszterkészlet

Akkumlátor	jelzők	Akkumlátor	jelzők
A	F	A'	F'
B	általános célú regiszterek	B'	C'
	C		
D	E	D'	E'
H	L	H'	L'
	megszakítás	memória	
	regiszter	frissítés	
	I	R	
	indexregiszter		
	IX		
	16 bit		
	indexregiszter		
	IY		
	16 bit		
	veremmutató		
	SP		
	16 bit		
	programszámláló		
	PC		
	16 bit		

3.3. Utasításkészlet

3.3.1. Címzés módok

A Z 80-as mikroprocesszor felépítése és utasításkészlete az adatok címzésének 7, az utasítások címzésének 4 módját teszi lehetővé.

Közvetlen adatcímzés

Az utasítás az operandust konstansként tartalmazza.

Abszolút címzés

Az utasítás az operandus helyét konstansként tartalmazza.

Indexregiszteres címzés

Az IX vagy IY regiszter tartalmához hozzáadja az utasításban szereplő egy bájtos előjeles számot { -128..127 }, és az így keletkezett szám alkotja az operandus memóriabeli címét.

Regisztercímzés

Az utasításban kódolt formában közvetlenül ki van jelölve az a regiszter

vagy regiszterpár, amelynek tartalma a műveletben operandusként vesz részt

Bennfoglalt címzés

Az utasítás az operandust automatikusan kijelöli, pl. a logikai műveletek egyik operandusa mindig az akkumulátor.

Indirekt regisztercímzés

Az utasításban kijelölt regiszterpár tartalma a művelet operandusának címét adja.

Bit címzés

Az utasításban egy $\{0..7\}$ intervallumba eső szám jelöli ki az operandusának azt a bitjét, amire a művelet vonatkozik.

Abszolút utasításcímzés

A következő végrehajtandó utasítás címe az utasításban található.

Relatív utasításcímzés

A következő végrehajtandó utasítás címét az utasításban található előjeles szám és a PC regiszter összege alkotja.

Indirekt (regiszterpáron keresztüli) utasításcímzés

A következő végrehajtandó utasítás címét a HL, IX, vagy IV regiszterpár tartalmazza.

Nullás lapú címzés

A következő végrehajtandó utasítás címét maga az operációs kód tartalmazza. Az RST utasítások használják, segítségükkel egy bájtós szubrutinhívást lehet végrehajtani a memória 0. lapjára, azaz az első 256 bájtra.

A lehetséges címek $8 \cdot n$ alakúak ahol $n = \{0..7\}$

Az utasításkészlet ismertetésénél a következő jelöléseket alkalmaztuk

:=

Legyen egyenlő

A

Akkumulátor.

n

8 bites adat.

nn

16 bites adat.

r, r'

B, C, D, E, H, L, A regiszterek valamelyike, értékük a felsorolás sorrendjében 0, 1, 2, 3, 4, 5, 7.

dd

BC, DE, HL, SP regiszterpárok valamelyike, értékük a felsorolás sorrendjében 0, 1, 2, 3.

qq

BC, DE, HL, AF regiszterpárok valamelyike, értékük a felsorolás sorrendjében 0, 1, 2, 3.

pp

BC, DE, IX, SP regiszterpárok valamelyike, értékük a felsorolás sorrendjében 0, 1, 2, 3.

rr

BC, DE, IY, SP regiszterpárok valamelyike, értékük a felsorolás sorrendjében 0, 1, 2, 3.

b

bit szám.

p

0 és 56 között 8-cal osztható szám

n

8 bites előjeles szám (az IX és IY relatív címzéseken).

e

8 bites relatív cím (a relatív ugróutasításokban).

ccc

Valamelyik kombináció az NZ, Z, NC, PO, PE, P, M feltételek közül, értékük a felsorolás sorrendjében 0, 1, 2, 3, 4, 5, 6, 7.

cc

Valamelyik kombináció az NZ, Z, NC, C feltételek közül, értékük a felsorolás sorrendjében 0, 1, 2, 3.

:=:

Csere.

()

Indirekció (a zárójelben álló mennyiség az operandus helyét adja).
Az utasításkészlet leírásánál a következő információkat adjuk meg:

- az utasítás mnemonikját,
- hatásának jelképes leírását és rövid szöveges ismertetését
- az utasítás kódjának bináris kombinációját,
- az utasítás által állított jelzők listáját.

3.3.2. Adatmozgató utasítások

LD r, r' r:=r'

az r' regiszter tartalmát átviszi az r regiszterbe.

kód: 01 r r'

Nem állít jelzőt.

LD r,n r:=n

az n konstanst átviszi az r regiszterbe.

kód: 00 r 110

Nem állít jelzőt.

LD r, (HL) r:=(HL)

A HL regiszterpár által megcímzett memória tartalmát átviszi az r regiszterbe.

kód: 01 r 110

Nem állít jelzőt.

LD r,(IX+d) r:=(IX+d)

Az utasításban levő d előjeles szám és az IX összege által megcímzett memória tartalmát átviszi az r regiszterbe.

kód: 11011101

01 r 110

d

Nem állít jelzöt.

LD r,(IY+d) r:=(IY+d)

Az utasításban levő d előjeles szám és az IY összege által megcímezett memória tartalmát átvizsgálja az r regiszterbe.

kód: 11111101

01 r 110

d

Nem állít jelzöt.

LD (HL),r (HL):=r

Az r regiszter tartalmát átvizsgálja a HL regiszterpár által megcímezett memóriába.

kód: 01110 r

Nem állít jelzöt.

LD (IX+d),r (IX+d):=r

Az r regiszter tartalmát átvizsgálja az utasításban levő d előjeles szám és az IX összege által megcímezett memóriába.

kód: 11011101

01110 r

d

Nem állít jelzöt.

LD (IY+d),r (IY+d):=r

Az r regiszter tartalmát átvizsgálja az utasításban levő d előjeles szám és az IY összege által megcímezett memóriába.

kód: 11111101

01110 r

d

Nem állít jelzöt.

LD (HL),n (HL):=n

Az n konstans átvizsgálja a HL regiszterpár által megcímezett memóriába.

kód: 00110110

n

Nem állít jelzöt.

LD (IX+d),n (IX+d):=n

Az n konstans átvizsgálja az utasításban levő d előjeles szám és az IX index összege által megcímezett memóriába.

kód: 11011101

00110110

d

n

Nem állít jelzöt.

LD (IY+d),n (IY+d):=n

Az n konstans átvizsgálja az utasításban levő d előjeles szám és az IY index összege által megcímezett memóriába.

kód: 11111101

00110110

d

n

Nem állít jelzöt.

LD A.(BC) A:=(BC)

A BC regiszterpár által megcímezett bájtot átviszi az A regiszterbe.
kód: 00001010
Nem állít jelzöt.

LD A,(DE) A:=(DE)
A DE regiszterpár által megcímezett bájtot átviszi az A regiszterbe.
kód: 00011010
Nem állít jelzöt.

LD A,(nn) A:=(nn)
Az nn címen elhelyezkedő bájtot átviszi az A regiszterbe.
kód: 00111010
 n A cím alacsonyabb helyiértéke
 n A cím magasabb helyiértéke
Nem állít jelzöt.

LD (BC),A (BC):=A
Az A regiszter tartalmát átviszi a BC regiszterpár által megcímezett bájtba.
kód: 00000010
Nem állít jelzöt.

LD (DE),A (DE):=A
Az A regiszter tartalmát átviszi a DE regiszterpár által megcímezett bájtba.
kód: 00010010
Nem állít jelzöt.

LD (nn),A (nn):=A
Az A regiszter tartalmát átviszi az nn címen elhelyezkedő bájtba.
kód: 00110010
 n A cím alacsonyabb helyiértéke
 n A cím magasabb helyiértéke
Nem állít jelzöt.

LD A,I A:=I
Az I (megszakítás) regiszter tartalmát átviszi az A regiszterbe.
kód: 11101101
 01010111
 S=1 ha az I negatív
 S=0 ha az I nem negatív
 Z=1 ha az I tartalma nulla
 Z=0 ha az I tartalma nullától különböző
 H=0
 P/V=IFF2
 N=0
 C=nem változik

LD A,R A:=R
Az R (memóriafrissítés) regiszter tartalmát átviszi az A regiszterbe.
kód: 11101101
 01011111
értelemszerűen ugyanaz, mint LD A,I-nél (I helyébe R-t írva)

LD I,A I:=A
Az A regiszter tartalmát átviszi az I (megszakítás) regiszterbe.
kód: 11101101
 01000111
Nem állít jelzöt.

LD R,A R:=A
 Az A regiszter tartalmát átviszi az R (memóriafrissítés) regiszterbe.
 kód: 11101101
 01001111
 Nem állít jelzöt.

LD dd,nn dd:=nn
 Az nn konstans átviszi a dd regiszterpárba
 kód: 00dd0001
 n A konstans alacsonyabb helyiértéke
 n A konstans magasabb helyiértéke
 Nem állít jelzöt.

LD IX,nn IX:=nn
 Az nn konstans átviszi az IX regiszterpárba.
 kód: 11011101
 00100001
 n A konstans alacsonyabb helyiértéke
 n A konstans magasabb helyiértéke
 Nem állít jelzöt.

LD IY,nn IY:=nn
 Az nn konstans átviszi az IY regiszterpárba.
 kód: 11111101
 00100001
 n A konstans alacsonyabb helyiértéke
 n A konstans magasabb helyiértéke
 Nem állít jelzöt.

LD HL,(nn) HL:=(nn)
 Az nn címen található két bájtot átviszi a HL regiszterpárba. Az alacsonyabb címen található bájt kerül az L regiszterbe.
 kód: 00101010
 n A cím alacsonyabb helyiértéke
 n A cím magasabb helyiértéke
 Nem állít jelzöt.

LD dd,(nn) dd:=(nn)
 Az nn címen található két bájtot átviszi a dd regiszterpárba.
 kód: 11101101
 01dd1011
 n A cím alacsonyabb helyiértéke
 n A cím magasabb helyiértéke
 Nem állít jelzöt.

LD IX,(nn) IX:=(nn)
 Az nn címen található két bájtot átviszi az IX regiszterpárba.
 kód: 11011101
 00101010
 n A cím alacsonyabb helyiértéke
 n A cím magasabb helyiértéke
 Nem állít jelzöt.

LD IY,(nn) IY:=(nn)
 Az nn címen található két bájtot átviszi az IY regiszterpárba.
 kód: 11111101
 00101010
 n A cím alacsonyabb helyiértéke
 n A cím magasabb helyiértéke
 Nem állít jelzöt.

LD (nn),HL (nn):=HL
A HL regiszterpár tartalmát átviszi az nn címen található két bájtba.
Az L regiszter tartalma kerül az alacsonyabb címre.
kód: 00100010

n A cím alacsonyabb helyiértéke
n A cím magasabb helyiértéke
Nem állít jelzöt.

LD (nn),dd (nn):=dd
A dd regiszterpár tartalmát átviszi az nn címen található két bájtba
kód: 11101101
01dd0011

n A cím alacsonyabb helyiértéke
n A cím magasabb helyiértéke
Nem állít jelzöt.

LD (nn),IX (nn):=IX
Az IX regiszterpárt átviszi az nn címen található két bájtba.
kód: 11011101
00100010

n A cím alacsonyabb helyiértéke
n A cím magasabb helyiértéke
Nem állít jelzöt.

LD (nn),IY (nn):=IY
Az IY regiszterpárt átviszi az nn címen található két bájtba.
kód: 11111101
00100010

n A cím alacsonyabb helyiértéke
n A cím magasabb helyiértéke
Nem állít jelzöt.

LD SP,HL SP:=HL
A veremmutatóba betölti a HL regiszterpárt.
kód: 11111001
Nem állít jelzöt.

LD SP,IX SP:=IX
A veremmutatóba betölti az IX regiszterpárt.
kód: 11011101
11111001
Nem állít jelzöt

LD SP,IY SP:=IY
A veremmutatóba betölti az IY regiszterpárt.
kód: 11111101
11111001
Nem állít jelzöt

PUSH qq
A qq regiszterpár értékét a verembe teszi. A kisebb helyértékű regisztert az SP-2, a nagyobb helyértékű regisztert az SP-1 címre teszi, majd az SP regiszterpárt kettővel csökkenti.
kód: 11qq0101
Nem állít jelzöt.

PUSH IX
Az IX regiszterpár értékét a verembe teszi a PUSH qq utasításnál leírt módon.

kód: 11011101
11100101
Nem állít jelzöt

PUSH IY
Az IY regiszterpár értékét a verembe teszi a PUSH qq utasításnál leírt módon.

kód: 11111101
11100101
Nem állít jelzöt

POP qq
A qq regiszterpárba betölti a verem tetején lévő értéket. Az SP címen található bájtot a kisebb, az SP+1 címen található a nagyobb helyiértékű regiszterbe teszi, majd az SP regiszterpárt kettővel növeli.

kód: 11qq0001
Nem állít jelzöt.

POP IX
A verem tetején található értéket az IX regiszterpárba teszi a POP qq utasításnál leírt módon.

kód: 11011101
11100001
Nem állít jelzöt.

POP IY
A verem tetején található értéket az IY regiszterpárba teszi a POP qq utasításnál leírt módon.

kód: 11111101
11100001
Nem állít jelzöt.

EX DE,HL DE:=HL
A HL regiszterpár tartalmát kicseréli a DE regiszterpár tartalmával.
kód: 11101011
Nem állít jelzöt.

EX AF,AF' AF:=AF'
A fő regiszterkészletben lévő AF regiszterpár tartalmát kicseréli a második készletben található AF' regiszterpár tartalmával.
kód: 00001000
A jelzők megváltoznak az F' -nek megfelelően.

EXX BC:=BC'
DE:=DE'
HL:=HL'
A fő regiszterkészletben lévő BC,DE,HL regiszterpárok tartalmát rendre kicseréli a második készletben található megfelelőikkel.
kód: 11011001
Nem állít jelzöt.

EX (SP),HL (SP):=HL
A HL regiszterpár tartalmát kicseréli az SP által mutatott két bájttal (a verembe utoljára betett értékkel).
kód: 11100011
Nem állít jelzöt.

EX (SP),IX (SP):=IX
Az IX regiszterpár tartalmát kicseréli az SP által mutatott két bájttal (a verembe utoljára betett értékkel).

kód: 11011101
11100011
Nem állít jelzöt.

EX (SP), IY (SP) := IY
Az IY regiszterpár tartalmát kicseréli az SP által mutatott két bájjal
(a verembe utoljára betett értékkel).

kód: 11111101
11100011
Nem állít jelzöt.

LDI (DE) := (HL)
HL := HL+1
DE := DE+1
BC := BC-1

A HL regiszterpár által mutatott bájtot átviszi a DE regiszterpár által mutatott bájtba, majd megváltoztatja a regiszterpárok értékét.

kód: 11101101
10100000
S= nem változik
Z= nem változik
H= 0
P/V=1, ha BC-1 = nem 0
P/V=0, ha BC-1 = 0
N=0
C= nem változik

LDIR
Az LDI utasítást ismétli addig, amíg a BC nulla nem lesz.

kód: 11101101
10110000
Ugyanaz, mint LDI-nél kivéve, a P/V = 0 {utoljára BC=0}).

LDD (DE) := (HL)
HL := HL-1
DE := DE-1
BC := BC-1

A HL regiszterpár által mutatott bájtot átviszi a DE regiszterpár által mutatott bájtba, majd megváltoztatja a regiszterpárok értékét.

kód: 11101101
10101000
Ugyanaz mint LDI-nél.

LDDR
Az LDD utasítást ismétli addig, amíg a BC nulla nem lesz.

kód: 11101101
10111000
Ugyanaz mint, LDIR-nél

CPI A-(HL)
HL := HL+1
BC := BC-1

Az összehasonlítás úgy történik, hogy a (HL)-lél címzett memóriarekesz tartalmát kivonja az akkumlátor tartalmából, de az akkumlátor tartalma változatlan marad.

kód: 11101101
10100001
S=1, ha az eredmény negatív
S=0, ha az eredmény nem negatív
Z=1, ha A=(HL)

Z=0, ha A=nem (HL)
 H=1, ha van átvitel a harmadik és a negyedik bit között
 H=0, ha nincs átvitel a harmadik és a negyedik bit között
 P/V=1, ha BC-1=nem 0
 P/V=0, ha BC-1=0
 N=1
 C= nem változik.

CPIR

A CPI utasítást ismétli addig, amíg a BC nulla nem lesz, vagy az A egyenlő nem lesz a HL által mutatott bájtal.

kód: 11101101

10110001

Ugyanaz mint, CPI-nél.

CPD

A-(HL)

HL:=HL-1

BC:=BC-1

A HL regiszterpár által mutatott bájtot összehasonlítja az A regiszterrel. Az összehasonlítás a CPI utasításnál leírtak szerint történik.

kód: 11101101

10101001

Ugyanaz mint, CPI-nél.

CPDR

A CPD utasítást ismétli addig, amíg a BC nulla nem lesz, vagy az A egyenlő nem lesz a HL által mutatott bájtal.

kód: 11101101

10111001

Ugyanaz mint, CPI-nél.

3.3.3. Aritmetikai és logikai utasítások

8 bites aritmetikai és logikai utasítások

ADD A+r

A:=A+r

Az akkumlátor és a kijelölt regiszter összegét az akkumlátorba teszi.

kód: 10000 r

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

Z=1, ha az eredmény nulla

Z=0, ha az eredmény nem nulla

H=1, ha van átvitel a harmadik és a negyedik bit között

H=0, ha nincs átvitel a harmadik és a negyedik bit között

P/V=1, ha van túlsordulás

P/V=0, ha nincs túlsordulás

N=0

C=1, ha van átvitel

C=0, ha nincs átvitel

ADD A,n

A:=A+n

Az akkumlátorhoz hozzáadja a konstans értékét.

kód: 11000110

D

Ugyanaz, mint az ADD A,r-nél.

ADD A,(HL)

A:=A+(HL)

Az akkumlátorhoz hozzáadja a HL által kijelölt bájt értékét.

kód: 10000110

Ugyanaz, mint az ADD A,r-nél.

ADD A,(IX+d) $A:=A+(IX+d)$

Az akkumlátorhoz hozzáadja a IX+d által kijelölt bájt értékét.

kód: 11011101

10000110

d

Ugyanaz, mint az ADD A,r-nél.

ADD A,(IY+d) $A:=A+(IY+d)$

Az akkumlátorhoz hozzáadja a IY+d által kijelölt bájt értékét.

kód: 11111101

10000110

d

Ugyanaz, mint az ADD A,r-nél.

A továbbiakban az s betűt használjuk az r, n, (HL), (IX+d), (IY+d) összefoglaló megjelölésére. A következőkben felsorolt aritmetikai és logikai utasítások az ADD utasítástól csak az aláhúzással megjelölt helyen térnek el, ezért csak ezeket jelöljük.

ADC A,s $A:=A+s+C$

Az akkumlátor, a kijelölt regiszter és a C jelző összegét az akkumlátorba teszi.

kód: xx001xxx

Ugyanaz, mint az ADD A,r-nél.

SUB s $A:=A-s$

Az akkumlátorból kivonja az s értékét.

kód: xx010xxx

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

Z=1, ha az eredmény nulla

Z=0, ha az eredmény nem nulla

H=1, ha van átvitel a harmadik és a negyedik bit között

H=0, ha nincs átvitel a harmadik és a negyedik bit között

P/V=1, ha van túlsordulás

P/V=0, ha nincs túlsordulás

N=1

C=1, ha van átvitel

C=0, ha nincs átvitel

SBC A,s $A:=A-s-C$

Az akkumlátorból kivonja az s és a C jelző értékét.

kód: xx011xxx

Ugyanaz, mint a SUB s-nél.

AND s $A:=A-s$

Az akkumlátor és az s közötti bitenkénti ES művelet, eredményt az akkumlátorba teszi.

kód: xx100xxx

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

Z=1, ha az eredmény nulla

Z=0, ha az eredmény nem nulla

H=1

P/V=1, ha a paritás páros
P/V=0, ha a paritás páratlan
N=0
C=0

XOR s A:=A-s

Az akkumlátor és az s közötti bitenkénti KIZARO VAGY művelet, eredményt az akkumlátorba teszi.

kód: xx101xxx

Ugyanaz, mint az AND s műveletnél, kivéve H:=0 (1 helyett)

OR s A:=A-s

Az akkumlátor és az s közötti bitenkénti VAGY művelet, eredményt az akkumlátorba teszi.

kód: xx110xxx

Ugyanaz, mint az AND s műveletnél, kivéve H:=0 (1 helyett)

CP s A-s

összehasonlítja az akkumlátor és az s-t, az eredmény nem kerül tárolásra. Az összehasonlítás a CPI utasításnál leírtakkal analóg módon kivonással történik.

Ugyanaz, mint az SUB s műveletnél.

Az s jelölés eddig volt érvényben, a továbbiakban a teljes kódot közöljük.

INC r r:=r+1

Az r regiszter tartalmát eggyel növeli.

kód: 00 r 100

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

H=1, ha az eredmény nulla

H=0, ha az eredmény nem nulla

H=1, ha van átvitel a harmadik és a negyedik bit között

H=0, ha nincs átvitel a harmadik és a negyedik bit között

P/V=1, ha r értéke 7FH volt az utasítás előtt

P/V=0, ha r értéke nem 7FH volt az utasítás előtt

N=0

C= nem változik

INC (HL) (HL):=(HL)+1

A HL című bájt tartalmát eggyel növeli.

kód: 00110100

értelemszerűen ugyanaz, mint INC r-nél {r helyett (HL)}.

INC (IX+d) (IX+d):=(IX+1)+1

A IX+d című bájt tartalmát eggyel növeli.

kód: 11011101

00110100

d

értelemszerűen ugyanaz, mint INC r-nél {r helyett IX+d}.

INC (IY+d) (IY+d):=(IY+d)+1

A IY+d című bájt tartalmát eggyel növeli.

kód: 11111101

00110100

d

95

CPL

Az akkumlátor értékének bitenkénti komplementjét képezi, azaz a nullákat egyesekké, az egyeseket nullákká alakítja.

kód: 00101111

S= nem változik

Z= nem változik

H=1

P/V = nem változik

N=1

C= nem változik

NEG

A:=0-A

Az akkumlátor előjelét megváltoztatja.

kód: 11101101

01000100

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

Z=1, ha az eredmény nulla

Z=0, ha az eredmény nem nulla

H=1, ha van átvitel a harmadik és a negyedik bit között

H=0, ha nincs átvitel a harmadik és a negyedik bit között

P/V=1, ha az akkumlátor tartalma 80H volt az utasítás előtt

P/V=0, ha az akkumlátor tartalma 80H volt az utasítás előtt

N=1

C=1, ha az akkumlátor nem nulla volt az utasítás előtt

C=0, ha az akkumlátor nulla volt az utasítás előtt

CCF

C:=1-C

A C jelző logikai komplementjét képezi.

kód: 00111111

S= nem változik

Z= nem változik

H= ismeretlen

P/V= nem változik

N=0

C=1, ha az utasítás előtt a C értéke 0 volt

C=0, ha az utasítás előtt a C értéke 1 volt

SCF

C:=1

A C jelzőt egyesre állítjuk

kód: 00110111

S= nem változik

Z= nem változik

H=0

P/V= nem változik

N=0

C=1

16 bites aritmetikai utasítások

ADD HL,dd HL:=HL+dd

A HL és a dd regiszterpár összegét a HL regiszterpárba írja.

kód: 00dd1001

S= nem változik

Z= nem változik

H=1, ha van átvitel a 11 és 12-es bitek között

H=0, ha nincs átvitel a 11 és 12-es bitek között

P/V= nem változik

N=0

C=1, ha van átvitel

C=0, ha nincs átvitel

ADD IX,pp IX:=IX+pp

Az IX és a pp regiszterpár összegét az IX regiszterpárba írja.

kód: 11011101

00pp1001

Ugyanaz, mint az ADD HL,dd utasításnál

ADD IY,rr IY:=IY+rr

Az IY és a rr regiszterpár összegét az IY regiszterpárba írja.

kód: 11111101

00rr1001

Ugyanaz, mint az ADD HL,dd utasításnál

ADC HL,dd HL:=HL+dd+C

A HL, a dd és a C összegét a HL-be írja.

kód: 11101101

01dd1010

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

Z=1, ha az eredmény nulla

Z=0, ha az eredmény nem nulla

H=1, ha van átvitel a 11 és a 12-es bitek között

H=0, ha nincs átvitel a 11 és a 12-es bitek között

P/V=1, ha van túlsordulás

P/V=0, ha nincs túlsordulás

N=0

C=1, ha van átvitel

C=0, ha nincs átvitel

SBC HL,dd HL:=HL-dd-C

A HL regiszterpárból kivonja a dd regiszterpárt és a C-t, az eredmény a HL regiszterpárba kerül.

kód: 11101101

01dd0010

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

Z=1, ha az eredmény nulla

Z=0, ha az eredmény nem nulla

H=1, ha van átvitel a 11 és a 12-es bitek között

H=0, ha nincs átvitel a 11 és a 12-es bitek között

P/V=1, ha van túlsordulás

P/V=0, ha nincs túlsordulás

N=1

C=1, ha van átvitel

C=0, ha nincs átvitel

INC dd dd:=dd+1

A dd regiszterpár tartalmát eggyel növeli.

kód: 00dd0011

nem állít jelzöt

INC IX IX:=IX+1

A IX regiszterpár tartalmát eggyel növeli.

kód: 11011101

00100011

nem állít jelzöt

INC IY IY:=IY+1

A IY regiszterpár tartalmát eggyel növeli.

kód: 11111101
00100011
nem állít jelzöt

DEC dd dd:=dd-1
A dd regiszterpár tartalmát csökkenti eggyel.
kód: 00dd1011
nem állít jelzöt

DEC IX IX:=IX-1
A IX regiszterpár tartalmát csökkenti eggyel.
kód: 11011101
00101011
nem állít jelzöt

DEC IY IY:=IY-1
A IY regiszterpár tartalmát csökkenti eggyel.
kód: 11111101
00101011
nem állít jelzöt

3.3.4. CPU vezérlő utasítások

NOP
A CPU egy utasításciklus idejére nem csinál semmit.
kód: 00000000
nem állít jelzöt

HALT
A CPU megáll, ebből az állapotból csak megszakítás vagy RESET lépteti ki.
kód: 01110110
nem állít jelzöt

DI
Tiltja a megszakításokat.
kód: 11110011
nem állít jelzöt

EI
Engedélyezi a megszakításokat.
kód: 11111011
nem állít jelzöt

IM 0
A CPU beáll a 0. megszakítási módba.
kód: 11101101
01000110
nem állít jelzöt

IM 1
A CPU beáll a 1. megszakítási módba.
kód: 11101101
01010110
nem állít jelzöt

IM 2
A CPU beáll a 2. megszakítási módba.
kód: 11101101
01011110
nem állít jelzöt

3.3.5. Forгатó és eltoló utasítások

RLCA

Az A regiszter bitjeit forгатja balra, minden bit eggyel magasabb helyiértékre kerül, a legfelső (hetes) bit a legalsó bitbe és a C jelzőbe kerül.

kód: 00000111

S= nem változik

Z= nem változik

H=OP/V=nem változik

N=0

C=a művelet előtti akkumulátor 7.bitjének értéke.

RLA

Az A regiszter bitjeit forгатja balra, minden bit eggyel magasabb helyiértékre kerül, a legfelső (hetes) bit a C jelzőbe másolódik. A C előző értéke kerül az A legalsó bitjébe.

kód: 00010111

Ugyanaz, mint az RLCA utasítás esetén.

RRCA

Az A regiszter bitjei forгатja jobbra, minden bit eggyel alacsonyabb helyiértékre kerül, a legalsó bit a legfelső (hetes) bitbe és a C jelzőbe kerül.

kód: 00001111

S= nem változik

Z= nem változik

H= 0

P/V nem változik

N= 0

C= a művelet előtti akkumulátor 0. bitjének értéke.

RRA

Az A regiszter bitjeit forгатja jobbra, minden bit eggyel alacsonyabb helyiértékre kerül, a legalsó bit a C jelzőbe másolódik. A C előző értéke kerül az A legfelső (hetes) bitjébe.

kód: 00011111

Ugyanaz, mint az RRCA utasításnál.

RLC r

Az r bitjeit forгатja balra, minden bit eggyel magasabb helyiértékre kerül, a legfelső (hetes) bit a legalsó bitbe és a C jelzőbe másolódik.

kód: 11001011

00000 r

S= 1, ha az eredmény negatív

S= 0, ha az eredmény nem negatív

Z= 1, ha az eredmény zérus

Z= 0, ha az eredmény nem zérus

H= 0

P/V=1, ha az eredmény paritása páros

P/V=0, ha az eredmény paritása páratlan

N= 0

C= a művelet előtti regiszter 7.bitjének értéke.

RLC (HL)

Az (HL) bitjeit forгатja balra, minden bit eggyel magasabb helyiértékre kerül, a legfelső (hetes) bit a legalsó bitbe és a C jelzőbe másolódik.

kód: 11001011

00000110

Ugyanaz, mint az RLC utasításnál.

RLC (IX+d)

Az (IX+d) bitjeit forgatja balra, minden bit eggyel magasabb helyiértékre kerül, a legfelső (hetes) bit a legalsó bitbe és a C jelzőbe másolódik.

kód: 11011101

11001011

d

00000110

Ugyanaz, mint az RLC utasításnál.

RLC (IY+d)

Az (IY+d) bitjeit forgatja balra, minden bit eggyel magasabb helyiértékre kerül, a legfelső (hetes) bit a legalsó bitbe és a C jelzőbe másolódik.

kód: 11111101

11001011

d

00000110

Ugyanaz, mint az RLC utasításnál.

RL m

Az m bitjei forgatja balra, minden bit eggyel magasabb helyiértékre kerül, a legfelső (hetes) bit a C jelzőbe másolódik. A c jelző értéke kerül a legalsó bitbe. Az m az r, (HL), (IX+d), (IY+d) valamelyikét jelöli. Az RLC utasításoktól az itt felsorolt egyéb forgató és eltoló utasítások csak az aláhúzott részekben térnek el, ezért csak ezt írjuk le.

kód: xx010xxx

Ugyanaz, mint az RLC utasításnál.

RRC m

Az m bitjeit forgatja jobbra, minden bit eggyel alacsonyabb helyiértékre kerül, a legalsó bit a legfelső (hetes) bitbe és a C jelzőbe másolódik.

kód: 001

Ugyanaz, mint az RRCA utasításnál.

RR m

Az m bitjeit forgatja jobbra, minden bit eggyel alacsonyabb helyiértékre kerül, a legalsó bit a C jelzőbe másolódik. A C előző értéke kerül a legfelső (hetes) bitbe.

kód: xx011xxx

Ugyanaz, mint az RRCA utasításnál.

SLA m

Az m bitjeit eltolja balra, minden bit eggyel magasabb helyiértékre kerül, a legfelső bit a C jelzőbe másolódik. A legalsó bitbe 0 kerül.

kód: xx100xxx

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

Z=1, ha az eredmény nulla

Z=0, ha az eredmény nem nulla

H=0

P/V=1, ha az eredmény paritása páros

P/V=0, ha az eredmény paritása páratlan

N= 0

C= művelet előtti adat 7.bitjének értéke.

SRA m

Az m bitjei eltolja jobbra, minden bit eggyel alacsonyabb helyiértékre kerül, a legalsó bit megőrzi előző tartalmát.

kód: xx101xxx

S=1, ha az eredmény negatív

S=0, ha az eredmény nem negatív

Z=1, ha az eredmény nulla

Z=0, ha az eredmény nem nulla

H=0

P/V=1, ha az eredmény paritása páros

P/V=0, ha az eredmény paritása páratlan

N= 0

C= művelet előtti adat 0. bitjének értéke.

SRL m

Az m bitjeit eltolja jobbra, minden bit eggyel alacsonyabb helyiértékre kerül, a legalsó bit a C jelzőbe másolódik. A legfelső bitbe 0 kerül.

kód: xx111xxx

Ugyanaz, mint az SRA m utasítás esetén.

A következő két utasítás hosszú BCD számokkal végzett műveletekhez alkalmas, nem hasonló az előző utasításokhoz, ezért ezek teljes kódját megadjuk.

RLD

Az A és a (HL) bitjeit forgatja balra, négyes csoportokban. Az A 7..4 bitjei helyben maradnak. A (HL) 3..0 bitjei a (HL) 7..4 bitjeibe, A (HL) 7..4 bitjei az A 3..0 bitjeibe, az A 3..0 bitjei a (HL) 3..0 bitjeibe kerülnek.

kód: 11101101

01101111

S= 1, ha az akkumulátor negatív a művelet után

S= 0, ha az akkumulátor nem negatív a művelet után

Z= 1, ha az akkumulátor nulla a művelet után

Z= 0, ha az akkumulátor nem nulla a művelet után

H= 0

P/V=1, ha az akkumulátor paritása páros a művelet után

P/V=0, ha az akkumulátor paritása páratlan a művelet után

N=0

C= nem változik.

RRD

Az A és a (HL) bitjeit forgatja jobbra, négyes csoportokban. Az A 7..4 bitjei helyben maradnak. Az A 3..0 bitjei a (HL) 7..4 bitjeibe, A (HL) 7..4 bitjei az (HL) 3..0 bitjeibe, az (HL) 3..0 bitjei a A 3..0 bitjeibe kerülnek.

kód: 11101101

01100111

Ugyanaz, mint az RLD utasításnál.

3.3.6. Bitműveletek

BIT b,r,

Az r b sorszámú bitjét teszteli, ha az a bit 1, akkor a Z jelző 0 lesz, ha az a bit 0, akkor a Z jelző 1 lesz.

kód: 11001011

01 b r

--

S= ismeretlen

Z= 1, ha a kiválasztott bit 0

Z= 0, ha a kiválasztott bit 1
H= 1
P/V= ismeretlen
N= 0
C= nem változik

BIT b, (HL)

A (HL) b sorszámu bitjét teszteli, ha az a bit 1, akkor a Z jelző 0 lesz, ha az a bit 0, akkor a Z jelző 1 lesz.

kód: 11001011

01 b 110

--

Ugyanaz, mint a BIT b,r utasításnál

BIT b, (IX+d)

A (IX+d) b sorszámu bitjét teszteli, ha az a bit 1, akkor a Z jelző 0 lesz, ha az a bit 0, akkor a Z jelző 1 lesz.

kód: 11011101

11001011

d

01 b 110

--

Ugyanaz, mint a BIT b,r utasításnál

BIT b, (IY+d)

A (IY+d) b sorszámu bitjét teszteli, ha az a bit 1, akkor a Z jelző 0 lesz, ha az a bit 0, akkor a Z jelző 1 lesz.

kód: 11111101

11001011

d

01 b 110

--

Ugyanaz, mint a BIT b,r utasításnál

SET b,m

Az m b sorszámu bitjét egyesbe állítja. Az m az r, (HL), (IX+d), (IY+d) valamelyikét jelöli. Az eltérés a BIT utasítástól csak az aláhúzott két bitben van, csak azt közöljük.

kód: 11xxxxxx

Nem állít jelzőt.

RES b,m

Az m b sorszámu bitjét nullába állítja. Az m az r, (HL), (IX+d), (IY+d) valamelyikét jelöli. Az eltérés a BIT utasítástól csak az aláhúzott két bitben van, csak azt közöljük.

kód: 10xxxxxx

Nem állít jelzőt.

3.3.7. Ugróutasítások

JP nn

PC:=nn

A programszámlálóba betölti az nn konstanst, azaz elugrik feltétel nélkül az nn címre.

kód: 11000011

n A cím alacsony helyértéke

n A cím magas helyértéke

Nem változtat jelzőt

JP cc,nn

Ha a ccc feltétel igaz elugrik az nn címre, egyébként a következő utasításon folytatja.

kód: 11ccc010

n A cím alacsony helyiértéke

n A cím magas helyiértéke

Nem változtat jelzöt

JR nn

Elugrik az nn címre. A címnek az utasítás körüli (-126..129) intervallumban kell lenni. Az utasításban csak a relatív címet tárolja, amit az $e := nn - PC$ összefüggésből számolhatunk. A PC már a következő utasításra mutat (ekkor már nagyobb kettővel)

kód: 00011000

e A relatív cím

Nem változtat jelzöt.

JR cc,nn

Ha a cc feltétel teljesül, akkor Elugrik az nn címre. A címnek az utasítás körüli (-126..129) intervallumban kell lennie. Az utasításban csak a relatív címet tárolja, amit az $e := nn - PC$ összefüggésből számolhatunk. A PC már a következő utasításra mutat. Ha a feltétel nem teljesül, akkor a következő utasításon folytatja.

kód: 001cc000

e A relatív cím

Nem változtat jelzöt.

DJNZ nn

$B := B - 1$

ha $B = \text{nem } 0$, akkor $PC := PC + e$

Csökkenti eggyel a B tartalmát. Ha az eredmény nem lett nulla, akkor elugrik az nn címre. A címnek az utasítás körüli (-126..129) intervallumban kell lenni. Az utasításban csak a relatív címet tárolja, amit az $e := nn - PC$ összefüggésből számolhatunk. A PC már a következő utasításra mutat.

kód: 00010000

e A relatív cím

Nem változtat jelzöt.

JP (HL)

Feltétel nélkül elugrik a HL regiszterpár által mutatott címre.

kód: 11101001

Nem változtat jelzöt.

JP (IX)

Feltétel nélkül elugrik az IX regiszterpár által mutatott címre

kód: 11011101

11101001

Nem változtat jelzöt.

JP (IY)

Feltétel nélkül elugrik az IY regiszterpár által mutatott címre

kód: 11111101

11101001

Nem változtat jelzöt.

3.3.8. szubrutinhívó és visszatérő utasítások.

CALL nn

A PC megnövelt (következő utasításra mutató) értékét a verembe teszi, majd elugrik az nn címre.

kód: 11001101

n A cím alacsonyabb helyiértéke

n A cím magasabb helyiértéke

Nem változtat jelzöt.

CALL ccc,nn

Ha a ccc feltétel igaz a PC megnövelt (következő utasításra mutató) értékét a verembe teszi, majd elugrik az nn címre. Ha ccc nem igaz, folytatja a végrehajtást a következő utasításon.

kód: 11cccc100

n A cím alacsonyabb helyiértéke

n A cím magas helyiértéke

Nem változtat jelzöt.

RET

A verem tetején található visszatérési címet a PC-be teszi.

kód: 11001001

Nem változtat jelzöt.

RET ccc

Ha a ccc feltétel igaz, a verem tetején található visszatérési címet a PC-be teszi.

kód: 11cccc000

Nem változtat jelzöt.

RETI

A verem tetején található visszatérési címet a PC-be teszi. A különbség a RET utasítástól az, hogy ezt az utasítást a Z 80 perifériák képesek felismerni.

kód: 11101101

01001101

Nem változtat jelzöt.

RETN

A verem tetején található visszatérési címet a PC-be teszi. A nem maszkolható megszakítást kezelő szubrutin végére kell tenni.

kód: 11101101

01000101

Nem változtat jelzöt.

RST p

Egy CALL utasítást hajt végre a p címre. A p, 0 és 56 közötti 8-cal osztható szám.

kód: 11 t 111 t=p/8

Nem változtat jelzöt.

3.3.9. Be és kiviteli utasítások.

IN A,(n)

Az A regiszterbe betölti az n című periféria tartalmát. A cím felső 8 bitjét az A regiszter előző tartalma képezi.

kód: 11011011

n A cím alsó 8 bitje

Nem változtat jelzöt.

IN r,(C)

Az r regiszterbe betölti az C regiszter által megcímzett periféria tartalmát. A cím felső nyolc bitjét a B regiszter előző tartalma képezi.

kód: 11101101

01 r 000

S=1, ha az input adat negatív
 S=0, ha az input adat nem negatív
 Z=1, ha az input adat nulla
 Z=0, ha az input adat nem nulla
 P/V=1, ha az input adat paritáson páros
 P/V=0, ha az input adat paritása páratlan
 N=0
 C=nem változik.

INI (HL):=(C)
 HL:=HL+1
 B:=B-1

A C című periférián található bájtot átviszi a HL regiszterpár által mutatott bájtba, majd megváltoztatja a regiszterpárok értékét. A perifériacím felső 8 bitjét a B regiszter adja.

kód: 11101101
 10100010
 S= ismeretlen
 Z= 1, ha B-1=0
 Z=0, ha B-1=nem 0
 H=ismeretlen
 P/V= ismeretlen
 N=1
 C= nem változik.

INIR
 Az INI utasítást ismétli addig, amíg a B nem nulla.

kód: 11101101
 10110010
 S= ismeretlen
 Z= 1
 H=ismeretlen
 P/V= ismeretlen
 N=1
 C= nem változik.

IND (HL):=(C)
 HL:=HL-1
 B:=B-1

A C című periférián található bájtot átviszi a HL regiszterpár által mutatott bájtba, majd megváltoztatja a regiszterpárok értékét. A perifériacím felső 8 bitjét a B regiszter adja.

kód: 11101101
 10101010
 Ugyanaz, mint az INI utasításnál.

INDR
 Az IND utasítást ismétli addig, amíg a B nem nulla.

kód: 11101101
 10111010
 Ugyanaz, mint az INIR utasításnál

OUT (n),A
 Az A regisztert kiviszi az n című perifériára A cím felső nyolc bitjét az A képezi.

kód: 11010011
 n A cím alsó 8 bitje
 Nem állít jelzöt.

OUT (C),r

Az r regisztert kiviszi az C című perifériára A cím felső nyolc bitjét az A képezi.

kód: 11101101

01 r 001

Nem állít jelzöt.

OUTI (C):=(HL)

HL:=HL+1

B:=B-1

A C című perifériára kiviszi a HL regiszterpár által mutatott bájtot, majd megváltoztatja a regiszterpárok értékét. A perifériacím felső 8 bitjét a B regiszter adja.

kód: 11101101

10100011

Ugyanaz, mint az INI utasításnál.

OTIR

Az OUTI utasítást ismétli addig, amíg a B nem nulla.

kód: 11101101

10110011

Ugyanaz, mint az INIR utasításnál.

OUTD (C):=(HL)

HL:=HL-1

B:=B-1

A C című perifériára kiviszi a HL regiszterpár által mutatott bájtot, majd megváltoztatja a regiszterpárok értékét. A perifériacím felső 8 bitjét a B regiszter adja.

kód: 11101101

10101011

Ugyanaz, mint az OUTI utasításnál.

OTDR

Az OUTD utasítást ismétli addig, amíg a B nem nulla.

kód: 11101101

10111011

Ugyanaz, mint az INIR utasításnál.

3.3.10. A jelzők beállítása

A műveletek eredményei állíthatják az F regiszter bitjeit az eredmény tulajdonságai szerint. A bitek elhelyezkedése a következő:

! S ! Z ! X ! H ! X ! P/V ! N ! C !

A nevük: S az előjel, Z a zérus, H a fél-átvitel, P/V a párosság és túlcsoordulás, N a kivonás vagy összeadás, C az átvitel. A két X jelző értéke nem meghatározott, jelentésük nincs. A következő táblázat foglalja össze, hogy melyik művelet melyik jelzöt állítja. A + jelöli azt, hogy a jelző változik a művelet eredményének megfelelően. A * azt jelenti, hogy a jelző nem változik. Az 1 vagy 0 esetén a jelző mindig ezt az értéket veszi fel. Ha X, akkor az eredmény nem definiált. A P/V sorában található P azt jelenti, hogy a jelző akkor 1, ha a művelet eredményében páros számú egyes bit van. Ha V-t találunk, ez az aritmetikai túlcsoordulást jelenti. Ha IFF jelölést találunk, akkor az megszakítás engedélyezés jelzője másolódott oda. Ha ez 1, akkor a mikroprocesszor figyelembe veszi a megszakításokat. A kivonás és a fél-átvitel jelzők csak a DAA utasítás-hoz szükségesek.

összefoglaló táblázat

! Művelet	! C	! Z	! P/V	! S	! N	! H
! ADD A,s :ADC A,s	! +	! +	! V	! +	! 0	! +
! SUB s :SBC A,s :CP s :NEG	! +	! +	! V	! +	! 1	! +
! AND s	! 0	! +	! P	! +	! 0	! 1
! DR s :XOR s	! 0	! +	! P	! +	! 0	! 0
! INC s	! *	! +	! V	! +	! 0	! +
! DEC s	! *	! +	! V	! +	! 1	! +
! ADD HL,ss :ADD IX,ss :ADD IY,ss	! +	! *	! *	! *	! 0	! X
! ADC HL,ss	! +	! +	! V	! +	! 0	! X
! SBC HL,ss	! +	! +	! V	! +	! 1	! X
! RLA :RLCA :RRA :RRCA	! +	! *	! *	! *	! 0	! 0
! RL m :RLC m :RR m :RRC m	! +	! +	! P	! +	! 0	! 0
! SLA m :SRA m :SRL m	!	!	!	!	!	!
! RLD :RRD	! *	! +	! P	! +	! 0	! 0
! DAA	! +	! +	! P	! +	! *	! +
! CPL	! *	! *	! *	! *	! 1	! 1
! SCF	! 1	! *	! *	! *	! 0	! 0
! CCF	! +	! *	! *	! *	! 0	! X
! IN r,C	! *	! +	! P	! +	! 0	! 0
! INI :IND :OUTI :OUTD	! *	! +	! X	! X	! 1	! X
! INIR :INDR :OTIR :OTDR	! *	! 1	! X	! X	! 1	! X
! LDI :LDD	! *	! X	! +	! X	! 0	! 0
! LDIR :LDDR	! *	! X	! 0	! X	! 0	! 0
! CPI :CPD	! *	! +	! +	! +	! 1	! X
! CPIR :CPDR	! *	! +	! 0	! +	! 1	! X
! LD A,I :LD A,R	! *	! +	! IPF	! +	! 0	! 0
! BIT b,s	! *	! +	! X	! X	! 0	! 1

a Z jelző akkor lesz 1, ha az eredmény 0 volt. Az S a művelet eredményének felső bitjét tartalmazza. A C, S, P/V jelzők megértéséhez foglalkozni kell a Z 80 számábrázolásával is.

Számábrázolási formák

A regiszterekben található számokat többféle módon is felfoghatjuk. Lehetnek előjeles, vagy előjel nélküli, BCD vagy bináris számok. Az előjeles számábrázolási formában a 8 bites regiszterek (-128..127), a 16 bites regiszterek (-32768..32767) intervallumokba eső számokat jelenthetnek. Ha a szám legfelső bitje 1, akkor ez negatív számot jelent. Ennek abszolút értékét úgy kaphatjuk meg, hogy az összes bitjeit negáljuk és hozzáadunk 1-et. 8 bites számokra ezt elvégzi a NEG utasítás. Az előjel nélküli számábrázolásnál a 8 bites számok (0..255), a 16 bites számok (0..65535) intervallumba eső értékeket képviselnek. A műveleti szabályok az előjel nélküli számokra ugyanazok, ezért a Z 80 maga ezt nem tudja, hogy mi minek akarjuk azt az értéket felfogni. A jelzőket viszont máshogy kell értelmezni a két esetben. Az S jelző az eredmény felső bitjét, azaz az előjeles szám előjelbitjét tartalmazza. A P/V az aritmetikai műveletekben (ahol V volt a bejegyzés) akkor áll 1-be, ha a művelet eredményeképpen tulcsordulás, hiba keletkezett. Ez olyankor lehet pl. ha két pozitív számot összeadva negatív eredményt kapunk:

$$96 + 32 = 01100000 + 00100000 = 10000000 = -128$$

Az eredmény felső bitje 1, ezért az eredmény negatív, hiba keletkezett.

A C jelző az előjel nélküli számoknál fontos. Akkor keletkezik, ha az előjel nélküli eredmény nem fér el az adott helyen pl:

$$100 + 200 = 01100100 + 11001000 = 00101100 = 44 \text{ Az eredmény hibás, ezt jelzi a C.}$$

A jelzőket felhasználhatjuk számok összehasonlítására. Ha CP utasítás után megnézzük a S Z és C biteket, azt találjuk, hogy az S akkor egy, ha az A-ban található szám kisebb, mint amihez hasonlítottuk (ekkor a -1 11111111 kisebb mint az 1). Ha a C=1, akkor az A előjel nélküli számként felfogva kisebb, mint amihez hasonlítottuk (ekkor az 1 =00000001 kisebb mint a 255 = 11111111, ami egyben a -1 előjeles formája is). A Z akkor egy, ha a művelet eredménye 0, azaz a számok egyformák. A C segítségével akármilyen hosszú számokkal végezhetünk aritmetikai műveleteket, mert az ADC és SBC utasítások figyelembe veszik, hogy az előző műveletben a szám nem fért el a rendelkezésre álló helyen, és egyet hozzáadnak vagy kivonnak még az eredményből, ha a C előző értéke 1. Így pl: 3 bájt hosszú számokat bájtanként úgy adunk össze, hogy a legalacsonyabb helyiértéken található bájtokat ADD, a többi növekvő helyiértékek felé haladva ADC utasításokkal adjuk össze. Legyen az egyik szám a B, D és E, a másik a C, H, és L regiszterekben:

```
LD A,L
ADD A,E
LD L,A
LD A,H
ADC A,D
LD H,A
LD A,C
ADC A,B
LD C,A
```


4. A Z 80 mikroprocesszor programozása ASSEMBLY nyelven

4.1. Az ASSEMBLY nyelv felépítése

Az ASSEMBLER a szimbolikus formában megadott utasításokat fordítja le a gép számára is érthető kódokká. Az elnevezése az angol összeszerelni szóból jött, a feladata is ez, csak a legkényelmetlenebb munkákat végzi el a programozó helyett, nem ad olyan kényelmes lehetőségeket, mint a magas szintű nyelvek.

A program sorokból áll, amit a képernyős szövegszerkesztő segítségével lehet megírni. Az ASSEMBLER csak az első 70 karaktert veszi figyelembe, a 70 karakter után következő bájtokat egészen a sorvége jeléig levágja. Így az utasítás értékes részét is elveszíthetjük, s ez fordítási hibát okoz. A sor négy fő mezőre tagolódik: címke, mnemonik, operandus és megjegyzésmezőre. A megjegyzésmező előtt egy ; (pontosvessző) áll, ez határolja el a sor többi részétől. A mezők mérete nem kötött, közöttük több betűköz vagy mezőhatároló (TAB) karakter állhat. Először a mnemonikmezővel foglalkozunk.

A mnemonikmezőbe írjuk az utasítás nevét (pl LD) vagy az ASSEMBLER számára szóló direktívákat. Először ezeket soroljuk fel.

4.1.1. Direktívák

ORG

A mögé írt szám lesz az elhelyezési számláló új értéke. Az ASSEMBLER ebben tartja nyilván, hogy a programban hol tart, mert szüksége van erre a címek és egyéb konstansok kiszámításánál. A programozó is felhasználhatja az értéket a \$ szimbólum segítségével. A fordítás megkezdésekor a \$ értéke 0.

A \$ használatára példa a következő utasítás, ami a HL-be betölti saját címét.

```
LD HL,$
```

LOAD

A mögé írt szám lesz a betöltési számláló új értéke. Az ASSEMBLER ebben tartja nyilván, hogy a készülő kódot hová teszi a memóriába. Általában megegyezik az elhelyezési számlálással, de nem szükségszerű, ezért lett a kettő megkülönböztetve. Így írhatunk pl. olyan programokat, amelyek az ASSEMBLER helyén fognak futni. Az elhelyezési számlálót a megfelelő értékre a betöltési számlálót egy szabad memóriaterületre irányítjuk. A programot végül a betöltési címre kell átvinni.

A fordítás megkezdésekor a betöltési számláló értéke 0. Az ORG direktíva után célszerű egy LOAD \$ direktívát írni.

EQU

Az előtte álló szimbolikus névhez hozzárendeli az utána álló kifejezés értékét. A kifejezésben minden szimbólumot előre kell definiálni.

```
pl: ALMA EQU 5
```

Ezentúl mindenhol, ahol ALMA-t írunk, az ASSEMBLER 5-öt fog helyette beírni.

```
pl: LD B,ALMA+2 ; a B regiszterbe 7 kerül.
```

Ha egy tetszőleges utasítás elé egy nevet írunk (ami után vagy, : vagy, szóköz áll), ez egyenértékű egy külön sorba leírt

```
NEV EQU $
```

sorral. A név azonos az előbb már említett címkével. A címke utáni kettőspont a más ASSEMBLEREK-hez szokott programozók részére maradt meg, használata nem kötelező.

```
ALMA: LD HL,ALMA
```

és

ALMA LD HL,ALMA
egyenértékűek.

DEFB

A mögötte álló (vesszővel elválasztott) kifejezéseket kiszámolja és beszeli egymás után a memóriába, minden bájtba egy értéket. A kifejezések értékének a (-128..255) intervallumba kell esni.

DEFW

A mögötte álló (vesszővel elválasztott) kifejezéseket kiszámolja és beszeli egymás után a memóriába, minden két bájtba egy értéket. Az alacsonyabb helyiértékű szám kerül az alacsonyabb címre. Az értéke (-32768..65535) között lehet.

DEFS

A mögötte álló (vesszővel elválasztott) kifejezés által meghatározott számú 0-val feltöltött bájtot ír be. Minden szimbólumnak már definiáltnak kell lenni.

ROM

A továbbiakban a készülő kódot betölti a memóriába, a DEFS direktívák helyére nullákat ír be. Ez az alapértelmezés a fordítás megkezdésekor.

RAM

A továbbiakban a kódok nem kerülnek be a memóriába, a DEFS direktíváknak csak a hosszát számolja, a DEFB, DEFW direktívák valamint utasítások használatakor hibajelzést kapunk. A RAM direktívához külön elhelyezési számláló tartozik, ezt is be kell állítani egy ORG direktívával. A RAM direktíva segítségével egyszerűen helyet foglalhatunk a használt változóinkkal anélkül, hogy ezzel a készülő kód hosszát növelnénk. pl:

```
LD A, (ALMA)
RAM
ORG 8000H
ALMA: DEFS 2
KORTE: DEFS 2
ROM
AND A
```

Igy a betöltő és az AND utasítás egymás után fog elhelyezkedni, míg az ALMA a 8000H, a KORTE a 8002H címen.

END

A fordítás befejezését jelenti. Használata nem kötelező. Ha van END, akkor ott van a program vége, ha nincs END, akkor a program utolsó utasítása után. Ezzel esetleg tagolhatjuk a programunkat, ha csak részeit akarjuk fordítani.

IF

Ha a mögötte álló kifejezés értéke nem 0, akkor a további sorokat fordítja, ha 0, akkor nem.

ELSE

Ha az előző IF direktíva után a kifejezés 0 volt, akkor most engedélyezi a fordítást.

ENDIF

Az IF és ELSE direktívák hatását megszünteti. Az IF - ELSE - ENDIF direktívák 6 mélységben egymásba ágyazhatók.

DEFL

Hasonlít az EQU direktívához, de nem kapunk hibajelzést, ha új értéket a-

dunk neki. A program különböző részeiben így ugyanaz a név más értéket hordozhat. Használata körültekintést igényel.

DEFM

Szöveg megadása: pl.

DEFM 'ez szöveg'

4.1.2. Utasítások

Az utasítások az előző fejezetben megjelölt utasításokat jelenti. Az utasítás nevét a mnemonikmezőbe, az operandusait az operandusmezőbe írjuk. Az utasítás elé címkét is írhatunk, így azonosítani tudjuk a program fontos részeit (szubrutin belépési pontja, programciklus eleje stb.). Az utasítások mögé érdemes a feladatukat magyarázó megjegyzéseket írni, hogy programunk érthetőbb legyen.

4.1.3. Aritmetikai kifejezések.

Az ASSEMBLER mindenhol, ahol számot vár, megért egy kifejezést is. A kifejezés tartalmazhat számokat, szimbólumokat, műveleti jeleket. A műveleti jelek között érvényesek a precedenciák, ezt zárójelezéssel bírálhatjuk felül. A számok lehetnek binárisak, oktálisak, decimálisak illetve hexadecimálisak. A bináris számok végén B betű áll. Az oktálisak végén vagy O vagy Q. A decimálisak végén állhat D, de ez nem kötelező, elhagyható a decimális számrendszerben. Hexadecimális számoknál a végén a H kötelező, a szám elején 0..9-nek kell állnia.

A legmagasabb prioritású a NOT, a logikai (bitenkénti) negálás jele. A következő a * (szorzás), / (osztás), MOD (maradék képzés), SHR (eltolás jobbra), SHL (eltolás balra), AND (bitenkénti és művelet). A legalacsonyabb prioritású a - (kivonás), + (összeadás), - (előjelcsere), OR (logikai vagy), XOR (logikai kizáró vagy) műveletek.

A - két operandus között állva kivonást, egy operandus előtt előjelváltást jelent. Egymagában álló + előjel nem megengedett, de nincs is értelme. A zárójelek 8 mélységben egymásba ágyazhatók.

Az azonosítók (szimbólumok) tetszőleges hosszúságúak lehetnek, de az első 8 karakterben különbözniük kell. Kötelezően betűvel kezdődnek és betűvel vagy számmal folytatódhatnak. A betű itt ékezet nélküli nagybetűt jelent.

4.1.4. Fordítást vezérlő speciális megjegyzések

Az ASSEMBLER a sor elején : 8 kettőssel kezdődő megjegyzéseket parancsnak veszi. A lehetséges betűk a következők:

L+

A listázás bekapcsolása.

L-

A listázás kikapcsolása. A listázási állapotot egy számláló jelzi, ezért n db L+ parancsot csak n db L- semlegesít.

S+

A szimbólumtáblát kiírja a fordítás végén. A szimbólum neve és értéke mögé még egy tájékoztató karakter is kerül. Ennek jelentése:

M

A címkét többször definiáltuk.

U

A címkét nem definiáltuk, de használni próbáltuk.

?

A címkét nem használtuk semmire, valami hibára figyelmeztet.

L

A címke értékét DEFL direktívával adtuk meg.

S-

Tiltja a szimbólumtábla kiírását. Az S+ vagy S- parancsok közül a legutolsó az érvényes.

U+

A nem használt szimbólumokat kiírja a fordítás végén.

U-

A nem használt szimbólumok tábláját nem írja ki. Itt is az utóljára megadott az érvényes.

E

A program belépési pontját adja meg. A MONITOR PC tárolójába írja az elhelyezési számláló aktuális tartalmát, így oda átlépve rögtön kezdetjük a futtatást.

I

Az utána következő nevű szöveget közvetlenül kazettáról befordítja az adott programsor mögé. A szöveget a szerkesztőből V paranccsal kell kivinni. A szövegnek hibátlanul kell lennie, mert a hiba esetén listázás alatt elveszhetnek a programsorok, mert a magnó nem áll meg. Hasonlóan nem célszerű bekapcsolni a listázást sem.

4.2. Az ASSEMBLER használata

4.2.1. Az ASSEMBLER elindítása

Az ASSEMBLER-t a szerkesztőből tudjuk elindítani, "Command" állapotban üttött Z segítségével. Kiíró:

Options:

Felirat után az érvényes fordítási paraméterek jelennek meg. Jelentésük:

L

A listázást engedélyezi. Egy ;\$L-parancs segítségével a szövegben letilthető.

P

A listázás a mátrix nyomtatón is megjelenik.

S

A szimbólumtáblát kiírja

U

A nem használt szimbólumokat kiírja

X

A fordítási hibáknál megáll

C

A készülő kódot a memóriába teszi

Az S, U, C, L, P parancsok a szöveg legelején ;\$ megjegyzésben megadott ugyanilyen paranccsal egyenértékűek.

Az options után az adott betű egyszeri megnyomása engedélyezi az adott funkciót, a következő tiltja. Ha már készen vagyunk a kiválasztással, akkor egy RETURN megnyomására indulhat a fordítás. Az első fordításkor a C és X be van kapcsolva, de ha átállítjuk, akkor a következő fordításnál már az általunk beállított opciókat kínálja fel.

A készülő lista az elhelyezési számláló értékét, a lefordított kódot, a forrásprogram szövegét és a hibajelzéseket tartalmazza. A hibás sort mindig listázza. Ekkor ha az X be van kapcsolva, akkor csak betűköz leütésére megy tovább a fordítás, egyéb karakterekre a szerkesztő jön vissza és a hibás sor elejére állítja a kurzort.

4.2.2. Hibajelzések

A fordítás közben előforduló hibajelzések a következők:

A

Hiba az operandusban, az utasítás nem érthető

C

Az utasításból készülő kód nem megengedett helyre kerülne. Ezt kapjuk akkor is, ha RAM direktíva után utasítások következnek.

E

Az IF - ELSE - ENDIF - direktívákkal van baj. (több ELSE, ENDIF IF nélkül stb.)

M

A címkét többször definiáltuk. A második definíciónál jelenik meg először.

O

A szám nagyobb, mint 65535.

Q

A sor végén valami szemét van. Elgépelte utasítás, vagy hiányzó ; okozhatja.

R

Az EQU direktíva után álló kifejezésben nem definiált szimbólum van.

U

A név ismeretlen.

V

Az érték nem megfelelő, túl nagy szám áll a relatív ugrás, egy bájtos konstans, vagy IX, IY relatív cím helyén.

5. Futtató - belövő: MONITOR

5.1. Regiszterkezelő

A MONITOR bejelentkezésekor a képernyő legalsó sorába regiszterek értéke írhatjuk. A számrendszer alapértelmezésben hexadecimális, de átváltható decimálisba is. A legfelső sorban a PC által mutatott utasítás és annak kódja látszik. A következő sorokban az SP, IX, IY, HL, DE, BC regiszterpárok és az HL, DE, BC regiszterpárok és a tartalmuk révén megcímzett memóriabájtok jelennek meg. Az AF regiszterpár után nem a tartalma révén megcímzett bájt értéke, hanem a jelzők értéke látható. Az AF alatt egy M: jel után a memória egy darabja látható, ezt az M: "regisztert" nevezzük memóriamutatónak. A MONITOR sok parancsának megadásakor számokat is be kell gépelni, ezeken a helyeken téves megadás esetén SHIFT+Balra törli az utolsó jegyet, a BRK pedig visszalép a regiszterkezelőbe. Ebben az állapotban a következő parancsok érvényesek.

(Betűköz) Egy utasítás végrehajtása a PC által mutatott címen. A végrehajtás után a regiszterek új értéke jelenik meg.

'='

Regiszterpár értékének beállítása. Ezután be kell gépelni a regiszterpár kezdőbetűjét ('P', 'S', 'X', 'Y', 'H', 'D', 'B', 'A', 'M'), majd a kívánt értéket.

','

Egy regiszter értékének beállítása. Ezután be kell gépelni a regiszter nevét ('H', 'L', 'D', 'E', 'B', 'C', 'A'), majd a kívánt értéket.

'*'

Jelző értékének beállítása. Ezután be kell gépelni a jelző nevét ('C', 'Z', 'S', 'P', 'H', 'N'), majd a kívánt értéket. Ez vagy 1, vagy 0 lehet.

','

Decimális és hexadecimális számrendszer között átvált.

','

Ha a decimális számrendszer érvényes, akkor a HL, DE, BC regiszterpárok kijelzését módosítja. Alapértelmezésben egy (0..65535) nagyságú szám jelenik meg, ./, hatására külön jelenik meg a két regiszter, azaz két (0..255) nagyságú számot látunk. Ujabb ./, hatására ismét egybe íródik a két regiszter.

','

Atugorja a következő utasítást. A PC a következő utasításra áll, más regiszterek értéke nem változik.

'+'

Ezután két számot vár, kiírja az összegüket és különbségüket.

'\$'

A visszafejtő programot hívja meg. Annak ismertetését lásd később.

'A'

Automatikus indítású gépi kódú programot ír ki. Meg kell adni neki a program kezdőcímét, hosszát, indítási címét és a nevét.

'B'

Töréspontokat lehet vele definiálni. Kiírja az eddigiek értékét, a nem használtak helyére ,----, kerül. Ha egy (1..4) tartományba eső számot

írunk be, akkor kéri az olyan sorszámú töréspont új értékét, egyébként kilép a parancsból. (A RESET megnyomása elrontja!!)

'C'

Memóriaterületet másol. Az adatokat honnan, hová, mennyit sorrendben kell megadni.

'E'

Program végrehajtása. A MONITOR-ba RET utasítással térhetünk vissza. Ha megadunk címet, onnan indít, ha nem adunk meg, akkor a PC értékétől kezdve. A töréspontok nem hatnak.

44

'F'

Memóriaterület feltöltése konstans tartalommal. Az adatokat hol, mennyit, mivel sorrendben kell megadni. A legutolsó szám csak a (0..255) tartományban lehet.

'G'

Program futtatása. A töréspontok hatásosak, a MONITOR-ba
RST 30H
utasítással térhetünk vissza.

'I'

Egy portról beolvas egy bajtot. A portcím csak (0..255) tartományban lehet.

'J'

Ha a PC által mutatott utasítás feltételes JP, JR, CALL, RET utasítás, akkor az ugrási címre tesz egy töréspontot és addig futtatja a programot. Ezzel léphetünk ki pl. ciklus közepéről. A ,B, paranccsal megadott töréspontokon is megáll.

'L'

Memóriaterület betöltése magnóról. Az információt a MONITOR ,W, paranccsával kell kiírni. Meg kell adni a betöltési címet és a program nevét.

'M'

A memóriakezelő meghívása. Erről majd később részletesebben szólunk.

'N'

Az utasítás mögé tesz egy töréspontot, és odáig futtat. Ezzel kerülhetjük el, hogy a szubrutin belsejébe lefusson, ha a CALL utasításnál ütjük le. A 'B' paranccsal megadott töréspontokon is megáll.

'O'

Egy portra kiír egy számot. A portcím és a szám (0..255) tartományba esik.

'O'

Visszatér a szövegszerkesztőbe.

'R'

Az SP által mutatott helyen talált címre tesz egy töréspontot, és addig

futtatja a programot. Szubrutinokból kilépésre használható. A ,B, paranccsal megadott töréspontokon is megáll.

'S'

Keresés a memóriában. Ha nem adunk meg keresési kezdőcímet utána, akkor az M: által meghatározott helytől kezdi keresni a régebben megadott bájtokat. A MONITOR-ba belépéskor egy db 0-t keres. Ha a címet megadjuk, akkor ezen címtől kezdve keres. Ha a cím után még írunk be bájtokat (maximum tizet), akkor azt fogja keresni. Az M: a megtalált minta helyére fog állni.

'T'

A szerkesztő szövegmezőjébe fejti vissza a megadott területet. Az adatokat kezdőcím, végcím sorrendben kell megadni. A parancs végrehajtása után automatikusan átlép a szerkesztőbe.

'V'

A 'W' paranccsal kiírt információt ellenőrzi.

'W'

Memóriaterület kiírása magnetofonra. Kezdőcímet, hosszt és nevet kell megadni.

'X'

A fő és a második regiszterkészletet cseréli ki.

5.2. Visszafejtő program

A visszafejtőt a regiszterkezelőből vagy a memória megjelenítőből hívhatjuk meg '\$' leütésével. Ha írunk utána számot, akkor attól a címtől kezdve, ha nem írunk, akkor az M: tartalmától kezdi a visszafejtést. A képernyőn megjelenik az ott talált 16 utasítás. A sorok elejére a cím, utána az utasítás kódja, majd az utasítás következik. Az ekkor érvényes parancsok a következők:

'Q', BRK

Visszatér a regiszterkezelőbe.

Lefelé nyíl

Egy utasítást lép a nagyobb címek irányába.

Felfelé nyíl

Egy utasítást lép a kisebb címek irányába.

'-'

16 utasítást lép a kisebb címek irányába.

'+'

16 utasítást lép a nagyobb címek irányába.

szám

A megadott címtől újra kezdi a visszafejtést.

'M'

Átlép a memória megjelenítőbe.

'!'

Decimális és hexadecimális számrendszer között atvált.

5.3. Memória megjelenítő

A memória megjelenítőt a regiszterkezelőből vagy a visszafejtőből hívhatjuk meg 'M' leütésével. Ha írunk utána számot, akkor attól a címtől kezdve, ha nem írunk akkor az M: tartalmától kezd a megjelenítést. A képernyő bal felső sarkába a kurzor címe kerül. A sorok közepén a bájtok tartalma hexadecimálisan, a jobb szélén ASCII formában jelenik meg. Decimális számrendszer esetén nincs karaktermező, mert nem fér el a képernyőn. A kurzor először a számmezőbe áll. Az itt érvényes parancsok a következők.

'Q' BRK

Visszatér a regiszterkezelőbe

Kurzormozgatók

A képernyőn megjelenített lapon belül mozgatják a kurzort

RETURN

A kurzor a bal felső sarokba áll. Itt beírt szám hatására a memóriát attól a címtől kezd el megjeleníteni

SHIFT+Jobbra

A szám és karaktermező között vált. A karaktermezőből ugyanezzel lehet visszatérni a számmezőbe. A karaktermezőben leütött karakterek kódja az adott helyre beíródik. A karaktermezőben ütött RETURN hatására is a bal felső sarokba áll a kurzor, de a cím megadása után a karaktermező bal felső sarkába áll vissza.

'-'

Az előző 128 bájtot jeleníti meg. A kurzor képernyő sarkától mért pozíciója megmarad.

'+'

A következő 128 bájtot jeleníti meg. A kurzor képernyő sarkától mért pozíciója megmarad.

'?'

Az ott talált két bájtot tekinti új megjelenítési kezdő címnek. A képernyő bal sarkában álló bájt címét úgy határozza meg, hogy a mostani és a következő cím különbsége osztható legyen 128-cal. Ezzel érhetjük el, hogy egy címen található memóriát egyszerűen megjeleníthessünk. Az 'X' paranccsal visszatérhetünk az előző pozícióba.

'R'

Az ott talált bájtot előjeles számnak veszi, majd hozzáadja a saját címe+1-t. Ez lesz az új megjelenítési kezdőcím. A képernyő bal sarkában álló bájt címét úgy határozza meg, hogy a mostani és a következő címkülönbsége osztható legyen 128-cal. A parancs segítségével a relatív ugróutasítások második bájtjára ráállva át tudunk lépni az ugrás helyére. Az 'Y' paranccsal visszatérhetünk az előző pozícióba.

'S'

Az előzőleg a regiszterkezelőben megadott mintát keresi a kurzor pozíciójától kezdve. A kurzor a megtalált mintára áll. Az 'X' paranccsal visszatérhetünk az előző pozícióba.

'X'

A '?' vagy 'S' parancs helyére térhetünk vissza.

'Y'

Az 'R' parancs helyére térhetünk vissza.

5.4. Szimbólikus belövési lehetőségek.

Az ASSEMBLER segítségével lefordított programokat egyszerűbben lehetjük be, ha a fordítás után azonnal a MONITOR-ba lépünk. Az ASSEMBLER szimbólumtáblája a szöveg után található, és ameddig nem sérül meg (valamilyen szövegjavító parancstól) addig a MONITOR fel tudja használni. A regiszterterítőnél a legfelső sorba az utasítás kódjának helyére egy szimbólum kerül, ha az utasítás tartalmaz 16 bites konstanst, és az ASSEMBLER szimbólumtáblájában van ilyen érték. Az összes helyen, ahol egy 16 bites számot vár a MONITOR, egy '?' leütésével megadhatunk egy szimbólumot is. Ilyenkor a legalsó sor törlődik, és egy

Symbol:

felirat után a legutoljára bevitt név és a hozzá tált érték íródik ki. Ha a névhez nincs érték, nem ír ki semmit. A gépelést RETURN fejezi be, az ekkor kiírt érték vagy 0 kerül arra a helyre ahol a számot gépelni kezdtük.